Libraries / Linking

Example

```
/* main.c */
#include "defs.h"

int buf[2] = {1, 2} ;

int main() {
   swap();
  return 0;
}
```

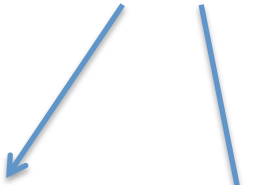
```
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap() {
  int temp;

  bufp1 = &buf[1];
  temp = *bufp0;
  *bufp0 = *bufp1;
  *bufp1 = temp;
}
```

gcc -O2 -g -o p main.c swap.c

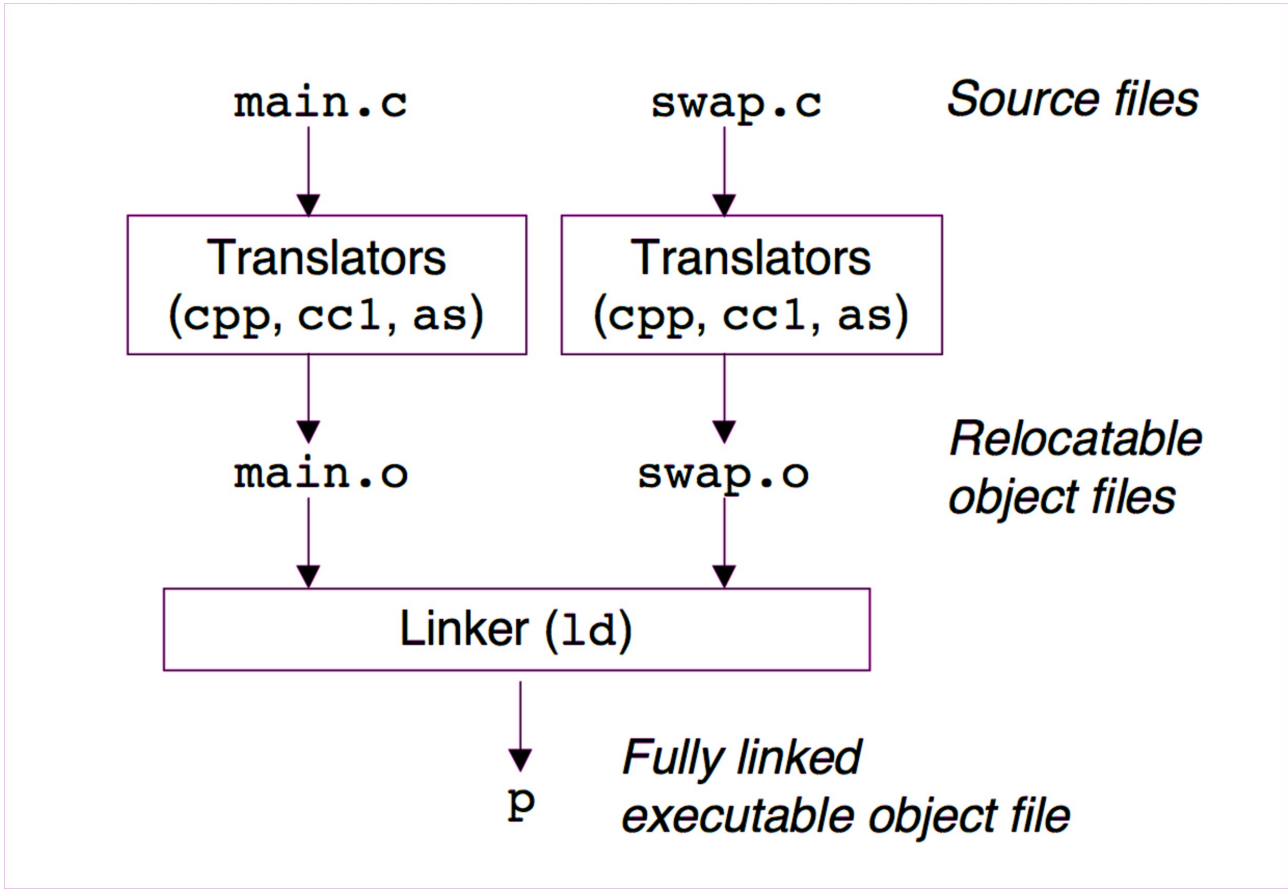


Code optimization

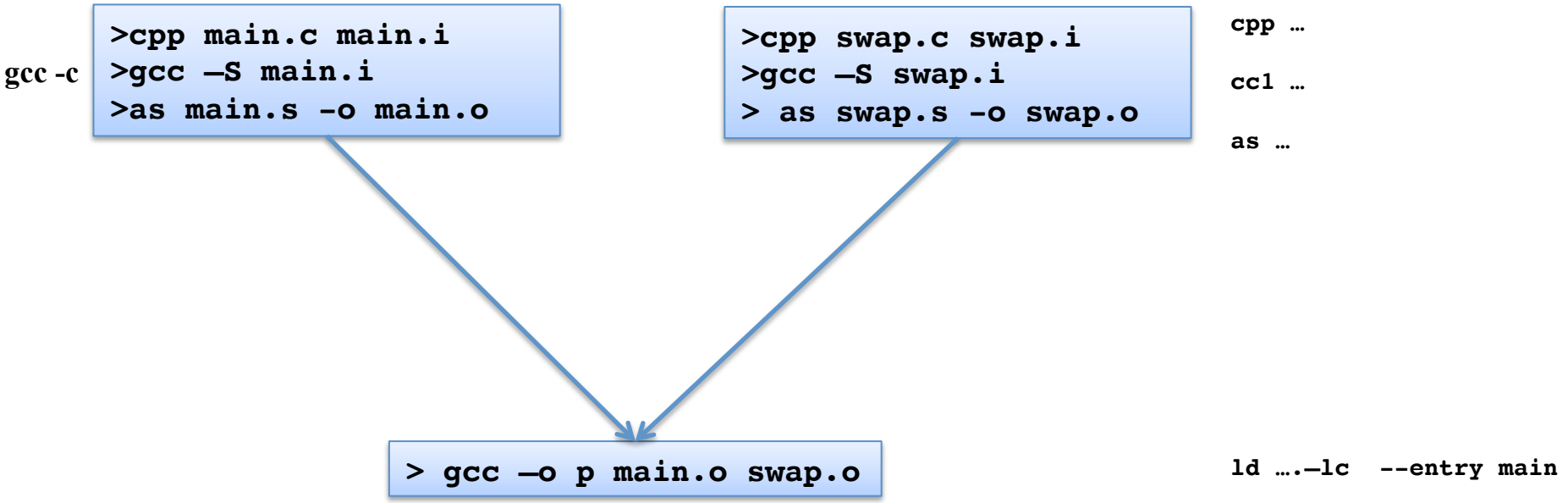
Produce debugging information (for use with debugger gdb)

```
/* defs.h */
void swap() ;
```

Static Linking



Compiling and Linking



Note:

- cpp is the C preprocessor
- cc1 is the internal command which takes preprocessed C-language files and converts them to assembly.
- as is the assembler
- ld is the linker

Object Files

- Relocatable object file. Contains binary code and data in a form that can be combined with other relocatable object files at compile time to create an executable object file.
- Executable object file. Contains binary code and data in a form that can be copied directly into memory and executed.
- Shared object file. A special type of relocatable object file that can be loaded into memory and linked dynamically, at either load time or run time.

- Object file formats vary from system to system
  - a.out : first Unix systems from Bell Labs
  - COFF (Common Object File format) : Early versions of System V Unix
  - PE (Portable Executable) : Windows NT
  - ELF (Executable and Linkable Format) : Modern Unix systems, such as Linux

Static Libraries (1)

- a static library or statically-linked library is a set of routines, external functions and variables which are resolved in a caller at compile-time and copied into a target application by a compiler, linker, or binder, producing an object file and a stand-alone executable

- To create a static library, or to add additional object files to an existing static library, use a UNIX command like this:

```
>ar rcs  libmy.a func1.o func2.o
```

- This sample command adds the object files func1.o and func2.o to the static library libmy.a, creating libmy.a if it doesn't already exist.

- To link with static library file:

```
>gcc main.o libmy.a
```

Static Libraries (2)

- To link with static library file in the LIBDIR directory:

```
>gcc main.o  ./LIBDIR/libmy.a   -o p.exe
```

- Using –L option (that indicates the directory where the library file is)

```
>gcc main.o  -L ./LIBDIR -lmy    -o p.exe
```

- If library is added to LIBRARY_PATH

```
>export  LIBRARY_PATH=$(LIBRARY_PATH):./LIBDIR
```

```
>gcc main.o   -lmy    -o p.exe
```

nm - list symbols from object files

```
>nm libmy.a

>nm func1.o

>nm p.exe
```

The characters that identify symbol types :

A : Global absolute symbol.
a : Local absolute symbol.
B : Global bss symbol.
b : Local bss symbol.
D : Global data symbol.
d : Local data symbol.
f : Source file name symbol.
L : Global thread-local symbol (TLS).
l : Static thread-local symbol (TLS).
T : Global text symbol.
t : Local text symbol.
U : Undefined symbol.

Shared (Dynamic) Libraries

- Shared libraries are libraries that are loaded by programs when they start.
- When a shared library is installed properly, all programs that start afterwards automatically use the new shared library.

```
>gcc -c -fpic  func1.c func2.c
>gcc -shared -o libmy.so func1.o func2.o
```

- You can link as follows

```
>gcc main.c ./libmy.so
```

- Or as follows:

```
>gcc main.c -L/home/canozturan/CMPE230/LIBRARY -lmy -o p.exe
>./p.exe
```

- Note: in order to be able to link as follows:

```
gcc main.c  -lmy -o p.exe
```

You have to add the library path to LIBRARY_PATH

```
>export LIBRARY_PATH=/home/canozturan/CMPE230/LIBRARY
```

and

```
>export LD_LIBRARY_PATH=/home/canozturan/CMPE230/LIBRARY
```

https://docs.freebsd.org/info/gcc/gcc.info.Environment_Variables.html

- See the documentation page "Environment Variables Affecting GNU CC" at:

gcc -print-search-dirs

- gcc complier's print-search-dirs option can print the directories searched

>**`gcc -print-search-dirs`**
install: /usr/lib/gcc/i486-linux-gnu/4.7/
programs: =/usr/lib/gcc/i486-linux-gnu/4.7/:/usr/lib/gcc/i486-linux-gnu/4.7/:/usr/lib/gcc/i486-linux-gnu/:/usr/lib/gcc/i486-linux-gnu/4.7/:/usr/lib/gcc/i486-linux-gnu/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../i486-linux-gnu/bin/i486-linux-gnu/4.7/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../i486-linux-gnu/bin/i386-linux-gnu/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../i486-linux-gnu/bin/
libraries: =/usr/lib/gcc/i486-linux-gnu/4.7/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../i486-linux-gnu/lib/i486-linux-gnu/4.7/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../i486-linux-gnu/lib/i386-linux-gnu/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../i486-linux-gnu/lib/../lib/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../i486-linux-gnu/4.7/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../i386-linux-gnu/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../lib/:/lib/i486-linux-gnu/4.7/:/lib/i386-linux-gnu/:/lib/../lib/:/usr/lib/i486-linux-gnu/4.7/:/usr/lib/i386-linux-gnu/:/usr/lib/../lib/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../../i486-linux-gnu/lib/:/usr/lib/gcc/i486-linux-gnu/4.7/../../../:/lib/:/usr/lib/

LD_LIBRARY_PATH

- LD_LIBRARY_PATH is an environment variable you set to give the run-time shared library loader (ld.so) an extra set of directories to look for when searching for shared libraries

- Suppose you copy executable p.exe to another directory (where libmy.so is not present) and try to run it there:

```
> ./p.exe
./p.exe: error while loading shared libraries: libmy.so:
cannot open shared object file: No such file or directory
```

- To resolve this problem, you have to add the path where the libmy.so is present to the LD_LIBRARY_PATH

```
>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/canozturan/CMPE230/LIBRARY
```

Where environment variables are used